EXHIBIT 10

FILED UNDER SEAL

Exhibit 10 to Sonos's Opposition to Google's Motion to Strike the Expert Reports of Dr. Sc
HIGHLY CONFIDENTIAL - SOURCE CODE - ATTORNEYS' EYES ONLY

Expert Report Material Google Seeks to Strike	Sonos's Infringement Contentio
Ex. 1, ¶128: I will generally refer to the list of media items provided by the YouTube cloud infrastructure for playback as the "Watch Next queue." As discussed in Google's internal documents, the WatchNext service accesses this Watch Next queue by making calls to the PlaylistDocumentService that in turn makes calls to the PlaylistService. *See*, *e.g.*, GOOG-SONOSWDTX-00039785 [Server], 88, 90-91. Ex. 1, ¶235: **Third**, the Watch Next queue provided by the YouTube cloud infrastructure (e.g., managed by the PlaylistDocumentService and PlaylistService Innertube servers and accessed by the WatchNext service) amounts to "a remote playback queue provided by a cloud-based computing system associated with a cloud-based media service" under the plain and ordinary meaning of that claim term in light of the Court's construction of "playback queue." Ex. 3, ¶46: Dr. Bhattacharjee's Rebuttal Report largely ignores these playback scenarios. It appears that Dr. Bhattacharjee does this to push the false narrative that a local playback queue runs the show for a YouTube Sender's playback. *See, e.g.*, Bhatta. Rebuttal Report, ¶¶60-70. But as I explained in my Opening Report, it is the YouTube cloud infrastructure –and specifically, the one or more Innertube servers that host the WatchNext, PlaylistDocumentService (PLDS), and PlaylistService services – that provides the list of media items that run the show for a YouTube Sender's playback (what I refer to as the "Watch Next queue"). *See, e.g.*, Schmidt Op. Report, ¶¶124-30, 152, 231-35, 248.	Pages 9-10: Each Cast-enabled computing of any one of the YouTube, YouTube Music, YouTube Kids app is programmed such that mode in which the Cast-enabled computing for playback of a remote playback queue (re "Watch Next" queue) provided by one or m (e.g., a "Watch Next," "InnerTube," or "MI associated with the YouTube, YouTube Mu YouTube Kids media service. The aforeme satisfies claim limitation 1.4. Pages 10-11: As established by the evidence user initiates local playback of user-selected a YouTube, YouTube Music, YouTube TV, service on a Cast-enabled computing device Cast-enabled computing device to become o playback of a playback queue referred to he Next" queue that is remote from the Cast-en device and will contain (i) a locator of at lea that was selected by the user for playback al for additional media items that were identifi YouTube service based on the user's selecti content and are seeded for playback after th content, which are referred to as "Autoplay" is evidenced by the data object called playlist_util.py →PlaylistFiller.playlist_vide Page 15: GOOG-SONOSWDTX-00039785 Squad – Server 01.05.2021] at 87-88 ("The PlaylistDocumentService provides a represe It calls into the PlaylistService for video IDs

1

Exhibit 10 to Sonos's Opposition to Google's Motion to Strike the Expert Reports of Dr. So
HIGHLY CONFIDENTIAL - SOURCE CODE - ATTORNEYS' EYES ONLY

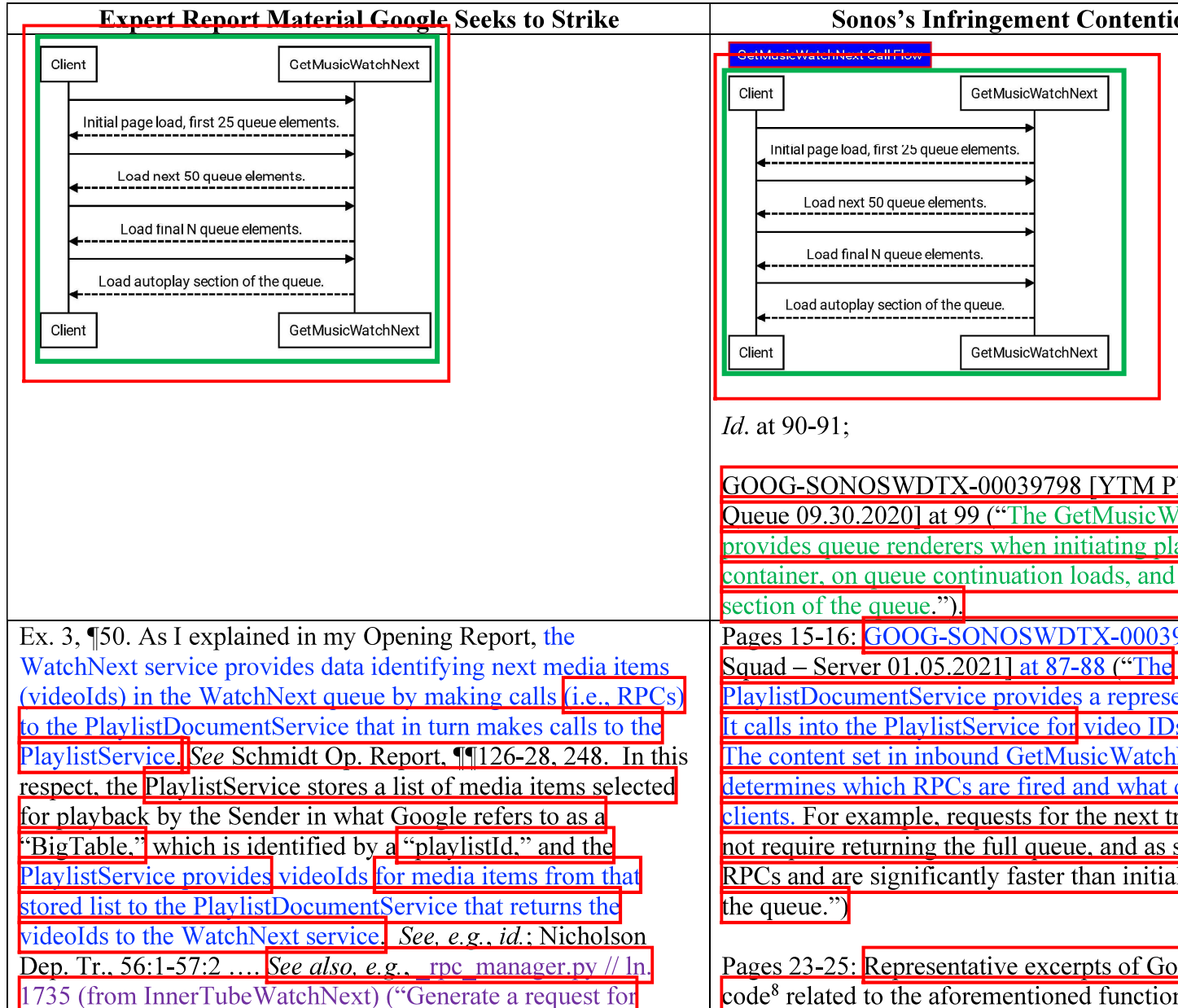| Expert Report Material Google Seeks to Strike	Sonos's Infringement Contentio
Ex. 3, ¶124: To start, that I am not aware of what label Google uses internally to refer to the list of media items provided by the YouTube cloud infrastructure for playback by a YouTube Sender is immaterial to whether I identified the "remote playback queue" that is called for by claim 1 of the '033 Patent. As I explained in my Opening Report (*see, e.g.,* Schmidt Op. Report, ¶¶99, 124-128, 231-42) and above (*supra* ¶¶46-52), the WatchNext, PLDS, and Playlist Services of the YouTube cloud infrastructure indisputably provide to a YouTube Sender a list of media items selected for playback and facilitate the YouTube Sender's playback through that list. For example, I cite to one of Google's documents that expressly explains that the PLDS "provide a representation of the queue" and gets videoIds in the queue from the PlaylistService. *See* GOOG-SONOSWDTX-00039785 [Server], 88. Moreover, as I explained before, Google and Dr. Bhattacharjee repeatedly represented to the Court that Google's accused YouTube systems **use only** a "remote playback queue" (or in Google's words, a "cloud queue"). *See, e.g.*, Schmidt Rebuttal Report, ¶302.	The content set in inbound GetMusicWatch determines which RPCs are fired and what clients. For example, requests for the next tr not require returning the full queue, and as s RPCs and are significantly faster than initia the queue.") Pages 23-25: Representative excerpts of Go code[8] related to the aforementioned function [*Source code trace for WatchNext service in* playlist_rpc_container.py → PlaylistRpcCo rpc_manager.py → get_playlist_doc_rpc() [FN8: Root directory: /2021-02- 02_YTServerInnerTubeWatchNext0929202
Ex. 2, ¶589: In contrast, as explained in my Opening Report, the infringing Watch Next queue that is provided by Google's YouTube cloud infrastructure is maintained by the YouTube cloud infrastructure while an infringing Sender operates in the mode where the Sender is configured to playback from the Watch Next queue. *See, e.g.*, Schmidt Op. Report, ¶¶126-28. In fact, the "PlaylistDocumentService" of the YouTube cloud infrastructure provides a representation of the queue that includes videoIDs that are for playback by the Sender. *Id.* The figure below shows the YouTube cloud infrastructure only sending a portion of the Watch Next queue to the Sender for playback, evincing the existence of a "remote playback queue":	Pages 15-16: GOOG-SONOSWDTX-00039 Squad – Server 01.05.2021] at 87-88 ("The PlaylistDocumentService provides a represe It calls into the PlaylistService for video ID The content set in inbound GetMusicWatch determines which RPCs are fired and what clients. For example, requests for the next tr not require returning the full queue, and as s RPCs and are significantly faster than initia the queue."), at 88:

2

Exhibit 10 to Sonos's Opposition to Google's Motion to Strike the Expert Reports of Dr. Sc
HIGHLY CONFIDENTIAL - SOURCE CODE - ATTORNEYS' EYES ONLY

Expert Report Material Google Seeks to Strike	Sonos's Infringement Contentic
	  Id. at 90-91; GOOG-SONOSWDTX-00039798 [YTM P Queue 09.30.2020] at 99 ("The GetMusicW provides queue renderers when initiating pla container, on queue continuation loads, and section of the queue.").
Ex. 3, ¶50. As I explained in my Opening Report, the WatchNext service provides data identifying next media items (videoIds) in the WatchNext queue by making calls (i.e., RPCs) to the PlaylistDocumentService that in turn makes calls to the PlaylistService. *See* Schmidt Op. Report, ¶¶126-28, 248. In this respect, the PlaylistService stores a list of media items selected for playback by the Sender in what Google refers to as a "BigTable," which is identified by a "playlistId," and the PlaylistService provides videoIds for media items from that stored list to the PlaylistDocumentService that returns the videoIds to the WatchNext service. *See, e.g., id.*; Nicholson Dep. Tr., 56:1-57:2 …. *See also, e.g.,* _rpc_manager.py // ln. 1735 (from InnerTubeWatchNext) ("Generate a request for	Pages 15-16: GOOG-SONOSWDTX-0003 Squad – Server 01.05.2021] at 87-88 ("The PlaylistDocumentService provides a represe It calls into the PlaylistService for video IDs The content set in inbound GetMusicWatch determines which RPCs are fired and what d clients. For example, requests for the next tr not require returning the full queue, and as s RPCs and are significantly faster than initial the queue.") Pages 23-25: Representative excerpts of Go code[8] related to the aforementioned function

3

Exhibit 10 to Sonos's Opposition to Google's Motion to Strike the Expert Reports of Dr. Sc
HIGHLY CONFIDENTIAL - SOURCE CODE - ATTORNEYS' EYES ONLY

Expert Report Material Google Seeks to Strike	Sonos's Infringement Contentio
playlist information from Playlist Document service."); playlist_document_service.proto // lns. 33-84 (handles a GetPlaylistDocumentRequest that can have a source set to "YTFE_WATCH_NEXT" indicating a request comes from a YT frontend WatchNext request for playback by a Sender): playlist_document_context.h // 41 ("Configures the request to playlist service and fetches the underlying YTPBPlaylist that the playlist document is going to be built out of."); playlist_service.h // lns. 78-79 ("The playlist stubby service implementation. A collection of RPCs for accessing and manipulating playlists stored in bigtable.").	*[Source code trace for WatchNext service in* _rpc_manager.py→get_playlist_doc_rpc() / [FN8: Root directory: /2021-02-02_YTServerInnerTubeWatchNext092920
Ex. 3, ¶51. For certain types of service-provided playlists, like radio- and mix-based playlists, the PlaylistService makes calls into another service for videoIds to add to the Watch Next queue. *See, e.g.*, GOOG-SONOSWDTX-00039785 [Server], 88; GOOG-SONOSWDTX-00039809 [WatchEndPoint]; Nicholson Dep. Tr., 49:10-50:19, 69:10-70:5, 73:13-74:9.	Page 11: GOOG-SONOSWDTX-00039785 Squad – Server 01.05.2021] Pages 15-16: GOOG-SONOSWDTX-00039 Squad – Server 01.05.2021] at 87-88 ("The PlaylistDocumentService provides a represe It calls into the PlaylistService for video ID The content set in inbound GetMusicWatch determines which RPCs are fired and what clients. For example, requests for the next tr not require returning the full queue, and as RPCs and are significantly faster than initia the queue.")
Ex. 3, ¶52: Google's YouTube Music Playback Squad documents titled "Server" and "The Queue" include some diagrams and flow diagrams illustrating the call flows between these services. *See* GOOG-SONOSWDTX-00039785 [Server], 88, 89, 90; GOOG-SONOSWDTX-00039798 [The Queue], 99. Mr. Nicholson explained that YouTube Music now just relies on the same WatchNext service as YouTube Main, so a more up-to-date, simplified diagram of the call flows between the	Pages 10-11: As established by the evidence user initiates local playback of user-selected a YouTube, YouTube Music, YouTube TV, service on a Cast-enabled computing device Cast-enabled computing device to become c playback of a playback queue referred to he Next" queue that is remote from the Cast-er device and will contain (i) a locator of at lea

Exhibit 10 to Sonos's Opposition to Google's Motion to Strike the Expert Reports of Dr. Sc
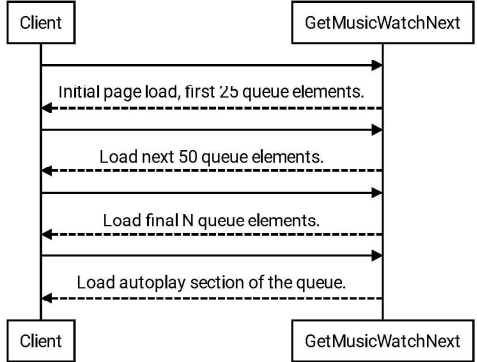**HIGHLY CONFIDENTIAL - SOURCE CODE - ATTORNEYS' EYES ONLY**

Expert Report Material Google Seeks to Strike	Sonos's Infringement Contentic
WatchNext, PlaylistDocumentService, and PlaylistService services in the YouTube cloud infrastructure is as follows, with a Sender interfacing with the WatchNext service:	that was selected by the user for playback al for additional media items that were identifi YouTube service based on the user's selecti content and are seeded for playback after the content, which are referred to as "Autoplay" is evidenced by the data object called playlist_util.py → PlaylistFiller.playlist_vide GOOG-SONOSWDTX-00039798 [YTM P Queue 09.30.2020] at 99 ("The GetMusicW provides queue renderers when initiating pl container, on queue continuation loads, and section of the queue.").] Pages 15-16: GOOG-SONOSWDTX-0003 Squad – Server 01.05.2021] at 87-88 ("The PlaylistDocumentService provides a represe It calls into the PlaylistService for video ID The content set in inbound GetMusicWatch determines which RPCs are fired and what clients. For example, requests for the next tr not require returning the full queue, and as RPCs and are significantly faster than initia the queue.") at 88:

5

Exhibit 10 to Sonos's Opposition to Google's Motion to Strike the Expert Reports of Dr. Sc

Expert Report Material Google Seeks to Strike	Sonos's Infringement Contenti
	GetMusicWatchNext Call Flow Client — GetMusicWatchNext Initial page load, first 25 queue elements. Load next 50 queue elements. Load final N queue elements. Load autoplay section of the queue. Client — GetMusicWatchNext *Id.* at 90-91; GOOG-SONOSWDTX-00039798 [YTM P. Queue 09.30.2020] at 99 ….
Ex. 3, ¶67. As another example, Google's documents explain that, while Casting, queue edit operations made by a user at a Sender result in the MDx session server making "calls to the PlaylistService to edit the underlying 'Remote Queue' playlist …." GOOG-SONOSWDTX-00039798 [The Queue], 800 ("The MDx Session Server manages the 'Remote Queue' playlist as well as the broader multi-device experience while Casting. Clients make queue edit operations (add to queue, re-order, remove from queue, etc) through the MDx Session Server, which makes appropriate calls to the PlaylistService to edit the underlying 'Remote Queue' playlist."); *see also, e.g.*, GOOG-SONOSWDTX-00041617 [YouTube Music Playback History in MDx Proposal], 20 ("The MDx Session Server adds the videoIds to the shared queue ('RQ' playlist) backed by the playlist service" and illustrating Session Server sending a "playlistModify" message to the Playlist Service), 24 ("MDx	Pages 10-11, 15-16 (citing and quoting from SONOSWDTX-00039798 [YTM Playback 09.30.2020]).

Expert Report Material Google Seeks to Strike	Sonos's Infringement Contenti
session server adds the videoEntries [(each containing a videoId)] to the shared queue ('RQ' playlist) via the Playlist Service" and illustrating Session Server sending a "playlistModify" message to the Playlist Service); GOOG-SONOSWDTX-00039827, 29, 32-33, 35	
Ex. 3, ¶69. Thus, the evidence undermines Dr. Bhattacharjee's opinions that the YouTube cloud infrastructure does not provide a list of media items for playback by a Sender before Casting and for playback by a Receiver after Casting. This is summarized best by the following illustration showing the WatchNext, PLDS, and PlaylistServices providing a list of media items for playback before and after Casting: [DIAGRAM] GOOG-SONOSWDTX-00039480 [Cast], 82 (annotations added).	Page 41 (citing GOOG-SONOSWDTX-000 Music Playback Squad – Cast])

7